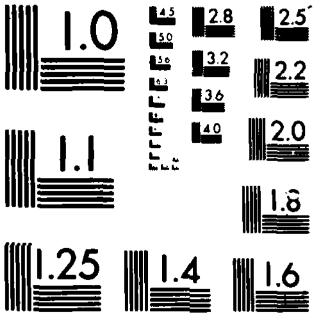
AD-A163 506 HARDWARE FEASIBILITY OF INFOPLEX DESIGN STRATEGY 1/1
REVISION(U) ALFRED P SLOAN SCHOOL OF MANAGEMENT
CAMBRIDGE MA CENTER FOR I.. H D TOONG ET AL. MAR 85
UNCLASSIFIED CISR-TR-18-REV N00039-81-C-0663 F/G 9/2 NL

END
FILMED



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

12

HARDWARE FEASIBILITY OF INFOPLEX DESIGN STRATEGY

H. D. Toong
Amar Gupta

Technical Report #18

Draft--March 1984
Revised--March 1985

Center for Information Systems Research
Massachusetts Institute of Technology
Sloan School of Management
77 Massachusetts Avenue
Cambridge, Massachusetts, 02139

Contract Number N00039-81-C-0663

Internal Report Number M010-8512-17

(12)

HARDWARE FEASIBILITY OF INFOPLEX DESIGN STRATEGY

H. D. Toong
Amar Gupta

Technical Report #18

Draft--March 1984

Revised--March 1985

Principal Investigator:
Professor S. E. Madnick

Prepared for:
Naval Electronics Systems Command
Washington, D.C.

REPORT DOCUMENTATION PAGE		**READ INSTRUCTIONS BEFORE COMPLETING FORM**
1. REPORT NUMBER Technical Report #18	2. GOVT ACCESSION NO. AD-A163506	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Hardware Feasibility of INFOPLEX Design Strategy		5. TYPE OF REPORT & PERIOD COVERED
		6. PERFORMING ORG. REPORT NUMBER M010-8503-18
7. AUTHOR(s) H. D. Toong, Amar Gupta		8. CONTRACT OR GRANT NUMBER(s) N00039-81-C-0663
9. PERFORMING ORGANIZATION NAME AND ADDRESS Center for Information Systems Research Sloan School of Management, M.I.T. Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Electronics Systems Command		12. REPORT DATE March 1985
		13. NUMBER OF PAGES 33
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Hardware feasibility, system resilience, fault tolerance capabilities, bus utilitzation, multiprocessing, multiprocessor bus

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Among its several design objectives, the key goals of INFOPLEX are to support an extremely large size of databases and to provide an unprecedented level of speed and efficiency in making database transactions. At another level, INFOPLEX attempts to dispense with the traditional approach of using monolithic mainframes. Instead, the design envisages use of arrays of relatively inexpensive microprocessors to allow for fast retrievals and to guarantee adequate fault tolerance capabilitites. These microprocessors need to be interconnected in an optimal manner to provide maximum system throughput. Concurrent multiprocessor operation represents one of the key ingredients of the hardware design strategy of INFOPLEX.

The communication bus is usually taken as the primitive for analyzing interconnection systems. By definition, the interconnection mechanism provides for transfer of instructions and/or data between the processing elements, the memory units and the peripherals. Estimates of bus usage in several current systems are reviewed.

A multiprocessor bus and multiprocessor operating system is presented and discussed.

In order to attain major improvements (in terms of storage capacity and speed of data retrievals) over conventional database systems, INFOPLEX strives to attain maximum concurrency at all dimensions. This high level of concurrency motivates the use of efficient split transaction protocols in order to support the inherent communication load involved. Overall, this report documents the feasibility, from the hardware viewpoint, of the INFOPLEX design strategy.

I. HARDWARE FEASIBILITY OF INFOPLEX DESIGN STRATEGY

1. Background

Among its several design objectives, the key goals of INFOPLEX are to support an extremely large size of database and to provide an unprecedented level of speed and efficiency in making database transactions. At another level, INFOPLEX attempts to dispense with the traditional approach of using monolithic mainframes. Instead, the design envisages use of arrays of relatively inexpensive microprocessors to allow for fast retrievals and to guarantee adequate fault tolerance capabilities. These microprocessors need to be interconnected in an optimal manner to provide maximum system throughput. Concurrent multiprocessor operation represents one of the key ingredients of the hardware design strategy of INFOPLEX.

In general, multiprocessor system design is motivated by one or more of the following factors:

(a) increased performance in terms of system throughput at a given level of component sophistication;

(b) increased ability to handle larger problems;

(c) increased system resilience and fault tolerance capabilities;

(d) inherent ease of enhancing system capabilities on a modular basis.

The present project attempts to develop a prototype system that will meet all the above objectives using contemporary microprocessors as primary building blocks. In developing an optimal state-of-the-art multimicroprocessor system, it is pertinent to identify and to evaluate design strategies adopted in other multiprocessor development projects.

The communication bus is usually taken as the primitive for analyzing interconnection systems. Buses may be separated into two generic types: dedicated and non-dedicated. Dedicated buses, permanently assigned either to one function or to one pair of modules, offer high throughput and simple protocols. Non-dedicated buses are more complex and permit more efficient usage of bandwidth. An example of such a bus is the single time-shared global bus which is discussed in Section 2.

By definition, the interconnection mechanism provides for transfer of instructions and/or data between the processing elements, the memory units and the peripherals. Estimates of bus usage in several current systems are summarized in Table 1 [Ref 1]. A single processor and a single memory can cause bus utilization to be as high as 70-80%. With more processors/memories, the bus becomes a performance bottleneck. Most designers have so far opted for multiple bus solutions. Depending on the geometry of such structures, the resulting network is termed a tree [Ref 2, 3], star [Ref 4], cube [Ref 5], hypercube [Ref 6, 7], hypertree [Ref 8], snowflake [Ref 9], cluster [Ref 10], and so on [Ref 11]. In all these cases,

Table 1. Bus Utilization in Current Systems

	8080*	8080**	6800	6502	9900	LSI-11
ADD8	82%	58%	92%	93%	55%	47%
SUB8	88%	58%	92%	93%	55%	47%
MULT8	71%	48%	66%	84%	39%	26%
DIV8	74%	50%	71%	80%	36%	20%
ADD16	82%	55%	92%	84%	55%	50%
SUB16	88%	58%	92%	84%	55%	50%
MULT16	74%	50%	75%	86%	39%	21%
DIV16	79%	53%	75%	83%	34%	20%
ADD32	81%	54%	92%	84%	48%	47%
SUB32	88%	58%	92%	84%	48%	47%
MULT32	87%	58%	82%	72%	37%	32%
DIV32	77%	52%	70%	77%	44%	37%
SORT8	77%	50%	57%	82%	45%	45%
SORT16	77%	52%	52%	80%	44%	44%
INTERRUPT	85%	57%	82%	60%	81%	46%

* 8080 uses bus during T1 (SYNCH) time.

** 8080 does not use bus during T1 (SYNCH) time.

there are many pairs of system resources with no direct link between each other. Any transfer of information between such pairs involves multiple transfer using intermediate nodes. This increases the total effort, introduces time delays, and wastes computing resources. In the case of microprocessors and other single chip elements (e.g., single chip memories and I/O interface units), each additional bus mandates more pins on the chip (unless one resorts to multiplexing with its inherent performance drawbacks), and hence increased costs (especially after 64 pins). It is therefore desirable to minimize the number of buses.

In all multiple bus systems (see Figure 1), the buses are interconnected to each other on a serial basis or a parallel basis or a hybrid thereof. Serial interconnection forms offer one and only one route between a given pair of system resources. Popular serial interconnection methods range from a single bus coupler to star and ring networks. Parallel interconnection systems offer multiple paths by using parallel buses, crosspoint switches or trunk system organization. The hybrid method attempts to minimize the average path-length between system resources and to maximize the number of parallel paths. The resulting structures, while providing excellent throughput from a theoretical viewpoint, can become too complex to be of immediate practical relevance. Table 2 [from Ref 6] summarizes the theoretical throughput of structures of varying complexity.

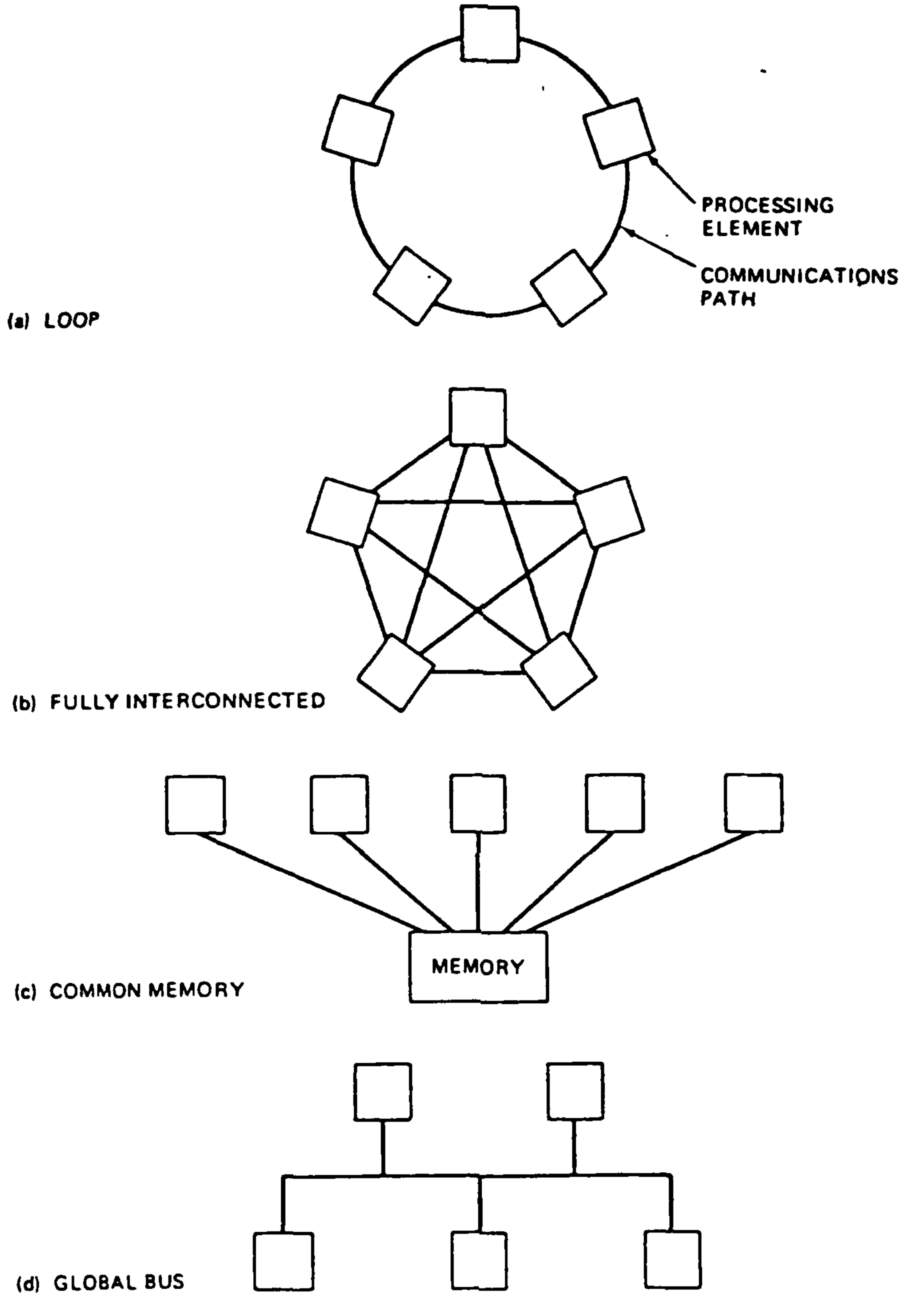
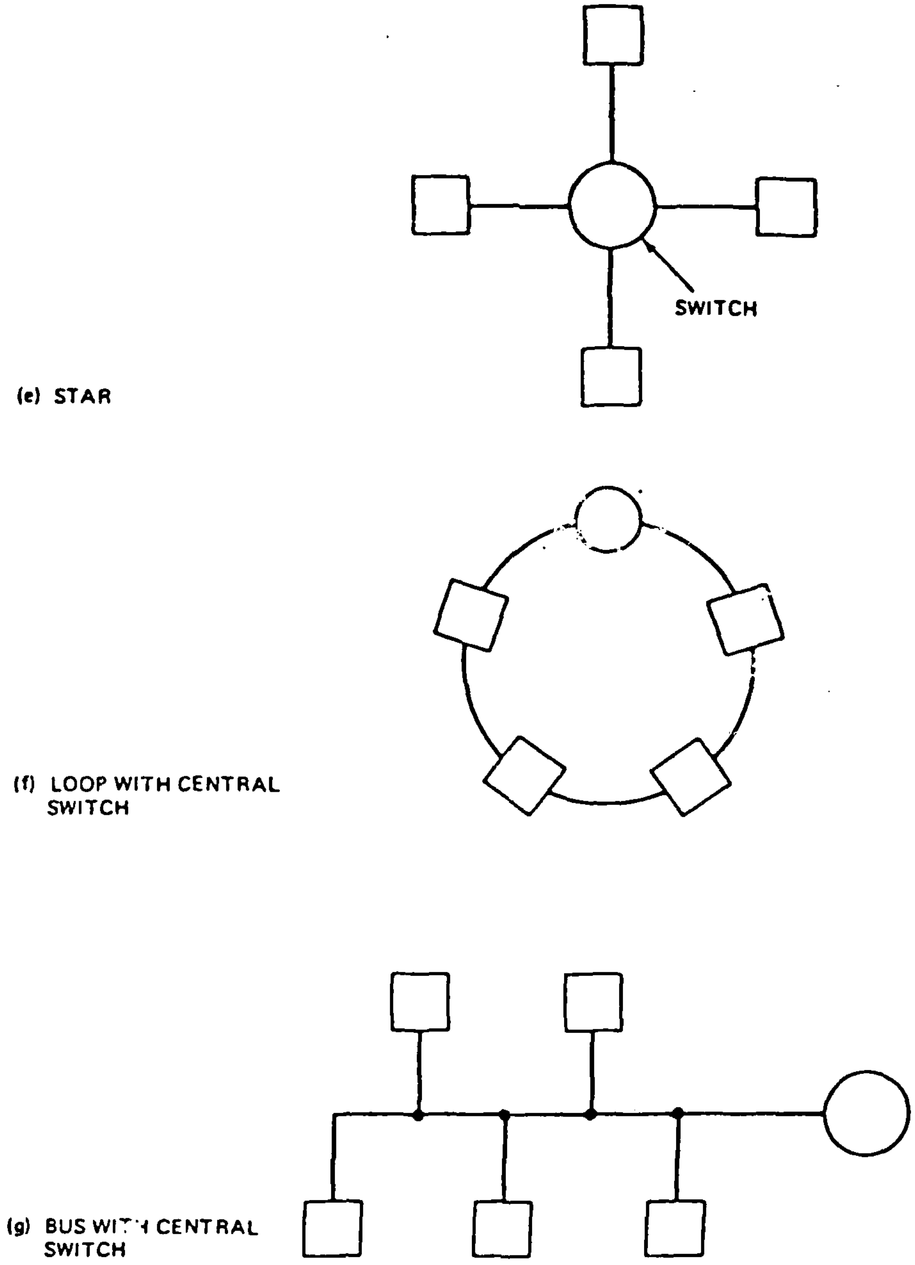


Figure 1. Typical Interconnection Methodologies

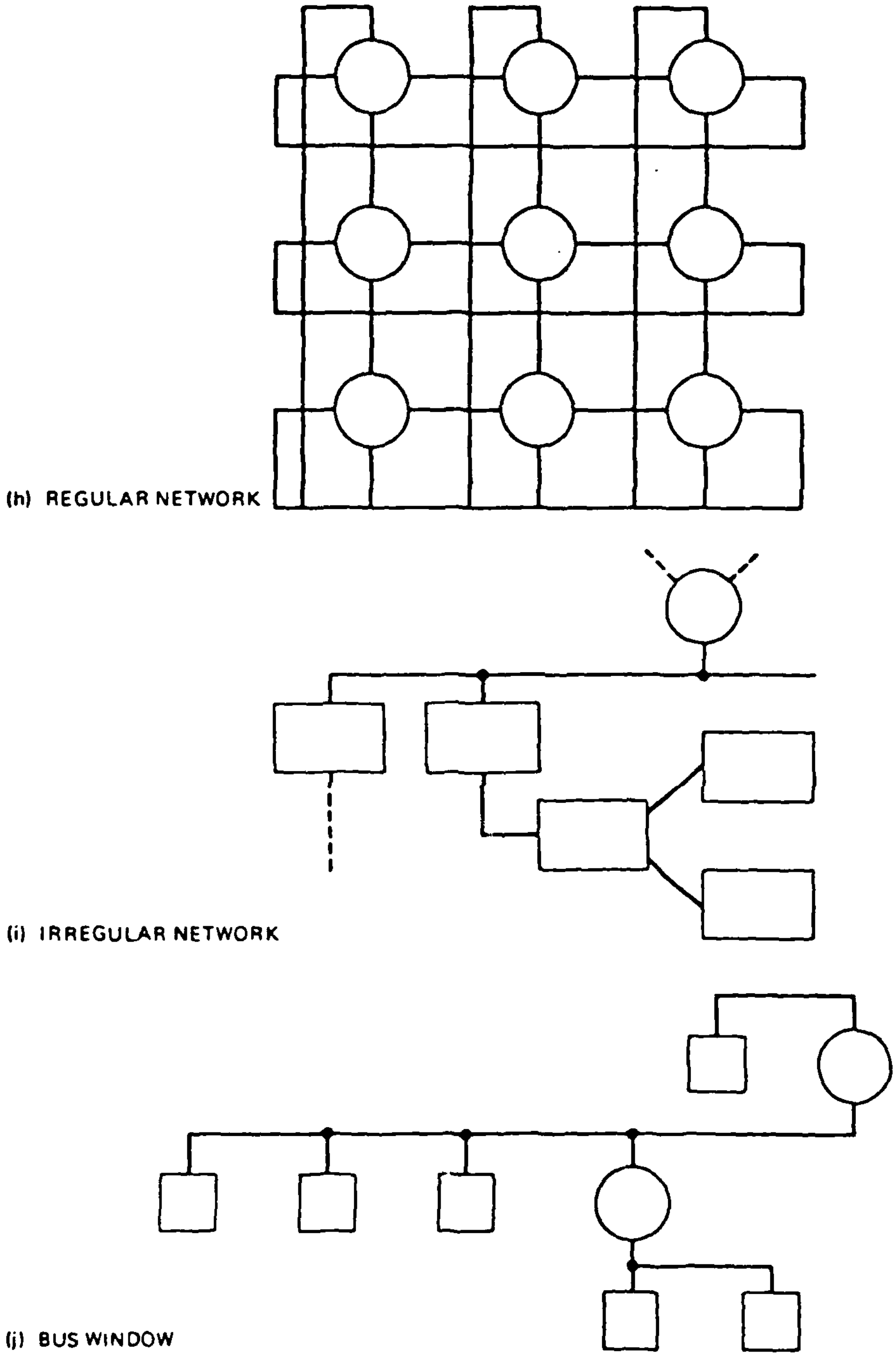


(e) STAR

(f) LOOP WITH CENTRAL SWITCH

(g) BUS WITH CENTRAL SWITCH

Figure 1. Typical Interconnection Methodologies (cont'd.)



(h) REGULAR NETWORK

(i) IRREGULAR NETWORK

(j) BUS WINDOW

Figure 1. Typical Interconnection Methodologies (cont'd.)

		DEPENDENCE OF CRITERIA UPON N ORDER OR DEPENDENCE :VALUE				RELIABILITY	
NETWORK STRUCTURE	NOTES ABOUT STRUCTURE	AVERAGE MESSAGE DELAY IN PATH USE TIMES	MESSAGE DENSITY IN MESSAGES PER LINK PER TIME UNIT	TOTAL CONNECTION COSTS PER NETWORK	CONNECTIONS PER NODE	WORST LOSS IF ONE POINT FAILS	TOTAL VALUE FOR LARGE NET SIZES N
One-Way Ring		N/2 N :0	N/2 N :0	2N N :6	2 Fixed :6	Paths Extra :2	14
Global Bus	(if N slots/bus)	N/2 N :0	N N :0	N N :6	1 Fixed :6	Busses Extra :2	14
Star	(1 hub) (N-1 arms)	2 Fixed:6	in hub N on arm 2 >N :1	2(N-1) N :6	at hub N-1 at arm 1 >N :1	Fail at hub Total:0	14
Completely Connected		1 Fixed:6	2/N Fixed:6	$(N^2-N)/2$ N^2 :0	N N :0	None :6	18
Tree (Fix Branch) ry Level)	(1 root) (B^{L-1} leaves) $N=(B^L-1)/(B-1)$	2(L-1) log N:4	root N leaf 2 >2N :1	2(N-1) N :6	B+1 Fixed :6	Fail at root Extra:2	19
Nearest Neighbor Mesh	($N=W^D$) (if D- fixed) (if W- fixed)	DW/4 $N^{1/D}$:2 log N:4	W/4 $N^{1/D}$:2 Fixed :6	2DN N :6 N log N:4	2D Fixed :6 log N :2	None :6 :6	22 22
Spanning Bus ($N=W^D$) Hypercube	(W slots in bus) (if D- fixed) (if W- fixed)	DW/2 $N^{1/D}$:2 log N:4	W $N^{1/D}$:2 Fixed:6	DN N :6 N log N:4	D Fixed :6 log N :2	None :6 :6	22 22
Dual-Bus Hypercube ($N=W^D$)	(if W slots/bus) (if low-use bus) (width W- fixed)	DW 2D log N:4	DW DW log N:4	2N 2N N :6	2 2 Fixed :6	None :6	26
Cube Connected Cycle	($N=D*2^D$ (dimension D)	7D/4 log N:4	5D/4 log N:4	3N N :6	3 Fixed :6	None :6	26

Table 2. Evaluation of Different Network Connection Structures with Respect to Reliability and to Order of Magnitude Changes in Average Message Delays, Message Traffic Densities, Total Network Connection Costs, and Connections Per Node as the Number of Nodes in the Network Increases to Very Large Values of "N". The Two Message Criteria Assume that Messages are Distributed Uniformly.

2. Single Time-Shared Bus

The various bus configurations analyzed in the previous section all used more than one bus to handle the interconnection communication loads. Well-known systems that use such multiple bus systems include (a) C.mmp (uses a 16x16 crosspoint switch); (b) Cm* (uses a three-level mechanism to support processor-memory, intra-cluster and inter-cluster communications); (c) MINERVA (at Stanford University). Although the use of additional buses increases the communication bandwidth, it also introduces delays, adds tasks relating to message-forwarding at intermediate nodes, and can reduce the overall throughput as well as reliability of the system. It is small wonder that none of these well-known projects led to popular commercial products.

The need for simplicity of design is even more relevant in the context of microprocessors and microprocessor-based systems such as INFOPLEX. The single time-shared bus (Figure 2) is the simplest multiprocessor structure. A number of modules (processors, memories and I/Os) share the bus on a time- or demand-multiplexed basis. A single bus communication involves the transfer of one word of data between modules and is called a transaction. When a device initiates a transaction it is called the master. After being granted control of the bus, the master supplies address and control information which identify the slave module and the type of transaction (read or write, memory or I/O). The trasaction is complete when the slave acknowledges the request, which may be done either explicitly through the use of an acknowledge signal, or implicitly according to a designed-in protocol.

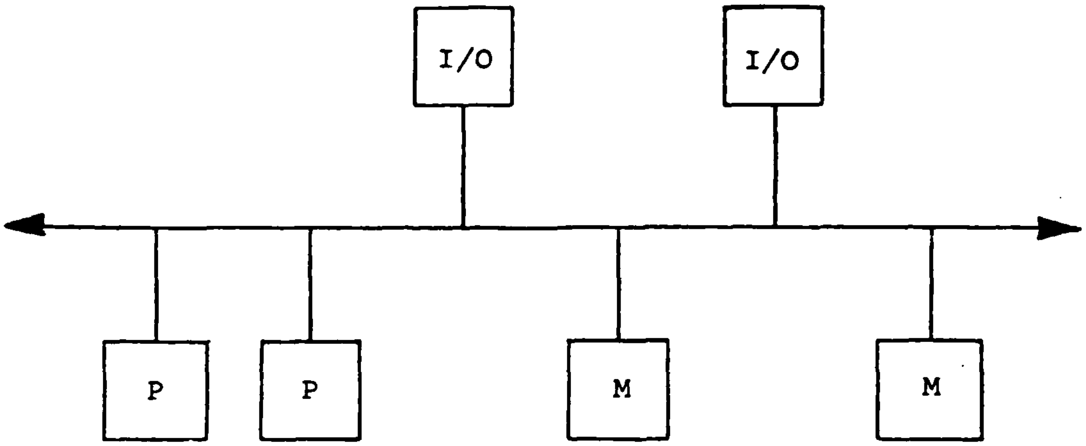


Figure 2. Single Time-Shared Bus

Permission to use the bus is generally granted in one of two major ways. Strict time multiplexing means that each potential master module is assigned a fixed-length time slot during which it may use the bus, on a rotating basis. If a module does not need to use the bus when its time arrives, the time slot is wasted. Demand multiplexing involves the use of a bus arbiter to grant bus access to one of the requesting modules whenever the bus becomes free. The manner in which the bus is assigned is called the arbitration algorithm. Demand-multiplexing eliminates the wasted time slot problem at the cost of introducing additional hardware. Nevertheless, most systems use demand multiplexing because of its greater efficiency. In the case of INFOPLEX, demand-multiplexing represents the favored strategy.

The single bus offers several advantages. First, there is total freedom of communication between modules using the bus. This universality means that the hardware enforces no logical partioning of the system, i.e., true multiprocessing is feasible. Second, no other interconnection scheme offers more minimal connectivity. Each module has only one port: its interface to the bus. The low complexity of a single-bus system makes it potentially economical. Third, the single bus offers flexibility, in the sense that it is easy to add or remove modules. This flexibility aspect would be very relevant in later phases of this project in order to implement systems with still higher throughput.

The primary deficiency of the single bus is that it offers only limited bandwidth, which is rapidly exhausted as more master modules are added to the system; thus the bus may become a communications bottleneck. Further, traditional single-bus schemes do not support concurrent communication; this means that only one master/slave pair can communicate at a time. Third, the existence of a central bus arbitrator may pose a reliability problem in the sense that its failure will be catastrophic. Last, the bus is limited both in its length and in the number of modules that may share it. These limitations are due to the fact that bus signals must travel the length of the bus, which has impact on the design of bus drivers and receivers and the speed of the bus protocol.

Traditional single-bus systems (e.g., UNIBUS) offer limited bandwidth, and as such the bus constituted the bottleneck. Innovative techniques, like split-transaction protocols, can be used to drastically increase the bus bandwidth and the overall throughput of such systems. In the case of INFOPLEX the following features are needed:
(a) Large Memory Addressing Capabilities;
(b) String Processing Capabilities;
(c) Ease of Multiprocessing;
(d) Ability to support mid-instruction interrupt.
In the next section we examine the features available on contemporary 16-bit microprocessors to support multiprocessing, especially on a split-transaction basis.

3. Microprocessor Characteristics

In order to achieve optimal system performance, it is critical to consider the specifications of contemporary chips in some detail. The major characteristics of a number of prominent 16-bit microprocessors are summarized in Table 3. We examine the multiprocessing support mechanisms available on four different 16-bit chips (the Intel 8086, the Z-8000, the MC68000 and the NS 16000) in the following paragraphs.

3.1 Intel 8086. The Multibus is the structure for interfacing Intel's 8080/85/86 products. It supports a one-megabyte address space. The 8289 bus arbiter controls Multibus accesses by multiple masters. The control lines are designed according to a master-slave concept: a master (processor) in the system takes control of the Multibus; then the slave device (I/O or memory), upon recognizing its address, acts upon the command provided by the master. An asynchronous handshaking protocol allows modules of different speeds to use the bus. Although the basic definition in the bus standard specifies only two types of units--bus masters and bus slaves--the system also can include "intelligent" slaves, which cannot control the bus, but put more processing power into the bus slave. Multiple masters can be connected in either a daisy chain priority scheme or in a parallel priority scheme.

Coordination features of the 8086 multiprocessor include:

- the 8289 bus arbiter, which decides which master may use the bus during the next cycle;

TABLE 3. Specifications of 16-bit microprocessors not including functions provided by co-processors or auxiliary chips

	TI 9900	Intel 8086	Zilog-Z8000	Motorola 68000	NS 16032
Year of Commercial Introduction	1976	1978	1979	1980	1982
No. of Basic Instructions	69	95	110	61	82
No. of General-Purpose Registers	16	14	16	16	8
Pin Count	40	40	48/40	64	48
Direct Address Range (Bytes)	64K	1M	48M*	16M/64M	16M
Number of Addressing Modes	8	24	6	14	9
Basic Clock Frequency	3MHz	5MHz (4-8MHz)	2.5-3.9MHz	5-8MHz	10MHz
System Structures					
Uniform Addressability	X	X	X	✓	✓
Module Map and Modules	X	X	X	X	✓
Virtual	X	X	X	X	✓
Primitive Data Types					
Bits	✓	X	✓	✓	✓
Integer Byte or Word	✓	✓	✓	✓	✓
Integer Double-Word	X	X	✓	✓	✓
Logical Byte or Word	✓	✓	✓	✓	✓
Logical Double-Word	X	X	X	✓	✓
Character Strings (Byte, Word)	✓	✓	✓	X	✓
Character Strings (Double-Word)	X	X	X	X	✓
BCD Byte	X	✓	✓	✓	✓
BCD Word	X	X	X	X	✓
BCD Double-Word	X	X	X	X	✓
Floating-Point	X	X	X	X	X
Data Structures					
Stacks	✓	✓	✓	✓	✓
Arrays	✓	X	X	X	✓
Packed Arrays	✓	X	X	X	✓
Records	✓	✓	✓	✓	✓
Packed Records	X	X	X	X	✓
Strings	X	✓	✓	X	✓

TABLE 3. Specifications of 16-bit microprocessors (cont'd.)

	TI 9900	Intel 8086	Zilog-Z8000	Motorola 68000	NS 16032
Primitive Control Operations					
Condition Code Primitives	✓	X	✓	✓	✓
Jump	✓	✓	✓	✓	✓
Conditional Branch	✓	✓	✓	✓	✓
Simple Iterative Loop Control	✓	✓	✓	✓	✓
Subroutine Call	✓	✓	✓	✓	✓
Multiway Branch	X	X	X	X	✓
Control Structure					
External Procedure Call	X	X	X	X	✓
Semaphores	X	✓	✓	✓	✓
Traps	✓	✓	✓	✓	✓
Interrupts	✓	✓	✓	✓	✓
Supervisor Call	✓	X	✓	✓	✓
Compatibility with other microprocessors	X	X	X	X	X

*6 segments of 8M each

- the bus lock signal, activated on execution of lock prefix instructions, blocking interrupts and requests by other processors until the lock sequence is completed;
- semaphore using the lock prefix in conjunction with the XCHG instruction;
- synchronization to an external event using a WAIT instruction and the test input signal;
- escape instruction allowing other processors to obtain an instruction and/or a memory operand from the host;
- two bidirectional request/grant lines, used to share the local bus between one host and two other processors via a handshake sequence--request, grant, release;
- the 8288 bus controller that outputs system bus signals compatible with Multibus.

3.2 Z8000. Two different multimicroprocessor mechanisms are possible on the Z8000. Zilog has designed a FIFO buffer communication module, which can run each processor as a separate system and pass messages back and forth through buffers to achieve total system communication. The processors are very loosely coupled, and any high-speed resource sharing is virtually impossible.

The second multiprocessor mechanism employs two signal pins called micro-in and micro-out (MI and MO) for implementation of a daisy-chained, software-controlled, global priority scheme. A processor examines the chain for busy condition (global resource allocation locked). If the bus is not busy, the processor places a request into the chain and then re-examines it after a settling delay (to prevent races). The result of the operation is reported with a flag handled in software. Thus, with an appropriate software driver, a single global locking scheme can be implemented. However, the time required to operate this locking mechanism rules out any high-speed communication.

Multimicroprocessor operation with the Z8000 is facilitated by the following features:

- four special, privileged "multimicro" instructions--MBIT, MREO, MRES, and MSET;
- pins for bus request, bus acknowledge, multimicro in, multimicro out, and segment trap;
- test and set instructions, TSET and TSEB;
- special output instructions;
- bus arbitration mechanisms;
- normal and system modes;
- provision for asynchronous Z-bus to Z-bus communication using the Z8038 FIO;
- simple external SSI logic to establish actual daisy-chain;
- semaphore using TSET (test and set) to synchronize software processes that require exclusive access to certain data cr instructions at one time;
- sharing of large memory by various processors under the memory management scheme.

On the Z8000, six op-codes have been reserved for extended instructicns to be used in conjunction with extended processing units (coprocessors).

3.3 MC68000. In a traditional Motorola 68000 environment, each processor has a local bus with local memory and peripherals. This capability has been enhanced on the 68010 (and the 68020) through insertion of a virtual-memory bus into the architecture. The 68010 will use a 16-bit external bus, and the 68020 will offer a 32-bit external bus. Both of these systems offer virtual memory support through an instruction interrupt and roll-back feature. In all 68000 systems, a global bus connects all local buses together through bus arbitration modules (BAMs). A processor is free to execute at full speed in its own bus space until it needs something from another processor's area, or until another device needs something from the former's domain. This is not a true multiprocessing system, but rather a connected group of individual microcomputer systems. Resources are not equally available to each processor. Any access involving the global bus takes longer than a simple local access. Access from the global bus back to a local bus is obtained through a DMA operation. There are no strictly global, shared resources, and the mechanism is suitable only for low, nonlocal access rates. Also, there is nothing to prevent several processors from making continuous accesses into one processor, effectively stopping that processor entirely. With the priority on the global bus fixed, a processor with the lowest priority may never get a global transaction started or completed.

MC68000 multimicroprocessor operation is facilitated by:

- bus arbitration modules (BAMs), which provide support in global bus multiprocessor design;
- the TAS (test and set) instruction;
- signals for bus request, bus grant, and bus grant acknowledge, which provides necessary input signals for arbitration purposes. Such arbitration requires some external hardware.

Interlocked multiprocessor communication is achieved through an indivisible read-modify-write cycle. For this purpose, the TAS instruction is used, and the address strobe is asserted throughout the cycle to inhibit other bus members from accessing the bus. The bus arbitration handles overlapped bus arbitration and transmission; however, it is not very powerful for multiple CPUs. The extended bus arbitration provided by Versabus is more powerful, but the inherent master/slave nature of its protocol presents a major bottleneck as the number of processors increases.

3.4 NS16000. The NS16000 series uses local buses and system buses. The local bus can connect the NS16032 CPU to the NS16081 floating-point unit, the NS16082 memory management unit, and the NS16203 DMA controller. The system bus is used for communications to other processors and global memory, and also to the bus arbiter and the interrupt control unit. The two buses communicate through "drivers" and "address latch" circuitry. It is too early to comment on specific system capabilities and potential bottlenecks of the National bus protocols.

3.5 Multiprocessing Overview. In all the 16-bit chips, support for multiprocessing is rather primitive, and one must consciously avoid the various pitfalls mentioned above. Among the other multiprocessors, the amount of resource sharing in the 8086 Multibus design is more restricted than that in the MC68000 local/global bus structure. Although all resources on the system bus can be accessed by any master, local bus resources are directly accessible only by the resident 8086. A further constraint imposed by Multibus is the fixed master-slave relationship of devices on the system bus, limiting interprocessor communication to the level of mailbox messages via global memory. Multibus, like the MC68000 local/global structure, is subject to saturation by high-priority devices. Individual transactions on Multibus are much faster than those on the MC68000 bus for two reasons: fetches from MC68000 local memory involve contention with the local processor, while Multibus global memory fetches do not; MC68000 inter-BAM communication adds two additional steps to the global memory access procedure. The Z8000 offers special signal pins, MI and MO, and four special instructions to support multiprocessing.

The newer 32-bit chips offer superior mechanisms to support multiprocessing. For example, the Intel iAPX 432 offers Packetbus, which uses split transaction concepts for more efficient bus utilization. Variable length (1 to 10 byte long) data messages are used for request/reply, and a 32-bit word can be transferred in 250 nanoseconds. Intel's own estimate of bus efficiency (Figure 3) indicates that their bus is useful for integrating only up to five processors. Obviously, even if a decision is made to use off-the-

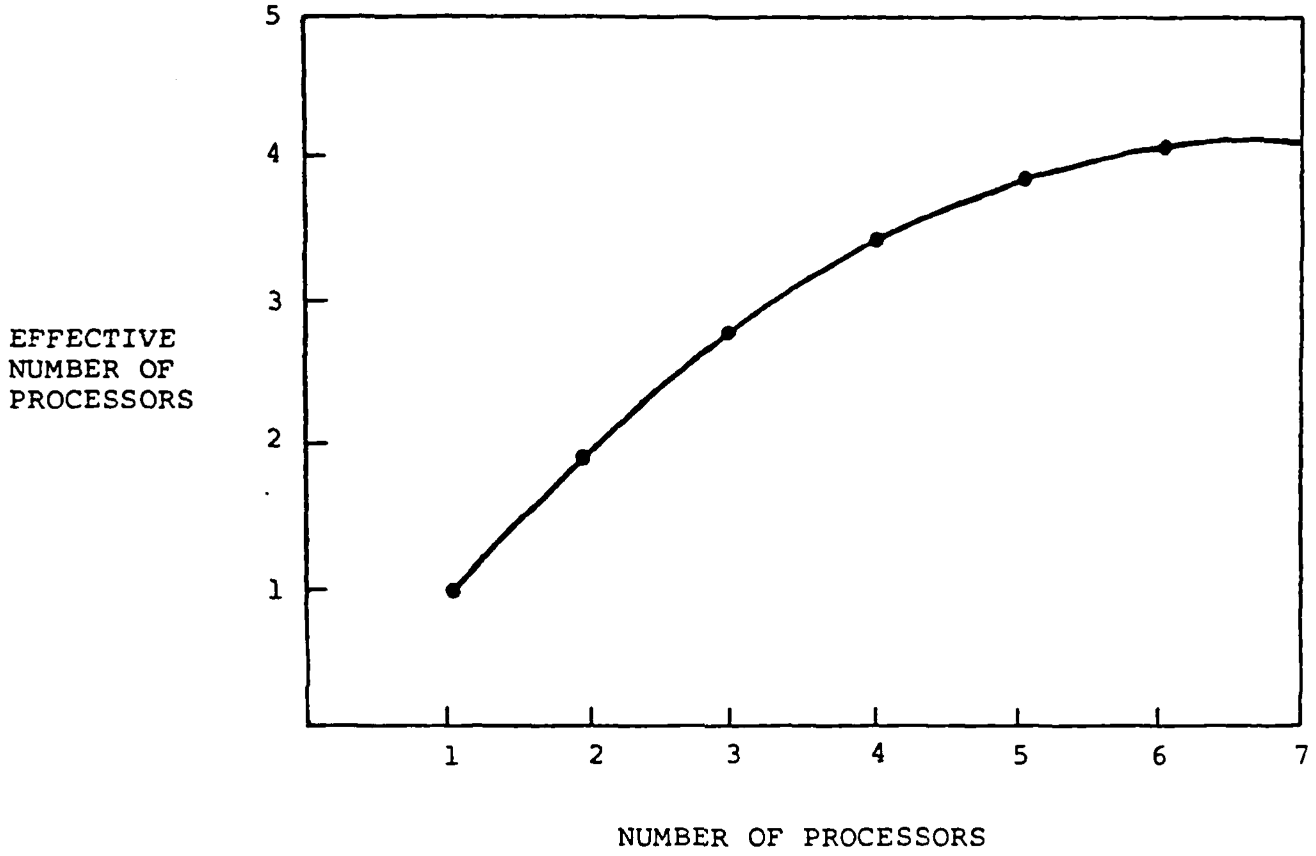


Figure 3. Intel iAPX 432 Bus Efficiency

shelf iAPX 432 (which is still not available in the debugged form), the Packetbus cannot be used in its standard form.

In order to implement a high performance system in the minimum time period, it is necessary to use a readily available and debugged microprocessor, and currently the MC68000 best meets these requirements. The bus structure must be implemented carefully to permit maximum bandwidth, using the MC68000 built-in facilities to the maximum feasible extent. With its 32-bit internal architecture, its powerful instruction set, and its support of a number of features that facilitate multiprocessing, the MC68000 represents, at the present time, the best hardware building block for implementing INFOPLEX.

4. Systems Overview

The previous section focused on alternative multiprocessing techniques and on the mechanisms available on current generation microprocessors to support multiprocessing. In this section, a blueprint of the INFOPLEX hardware architecture is outlined. The principal design objectives are as follows:

(a) Performance aspect: It should offer high availability, high performance and non-stop service;

(b) Technology aspect: It should use off-the-shelf technology:

(c) Architecture aspect: The system architecture should facilitate system growth on a modular basis;

(d) Compatibility aspect: It should be compatible with XODIAC and AOS;

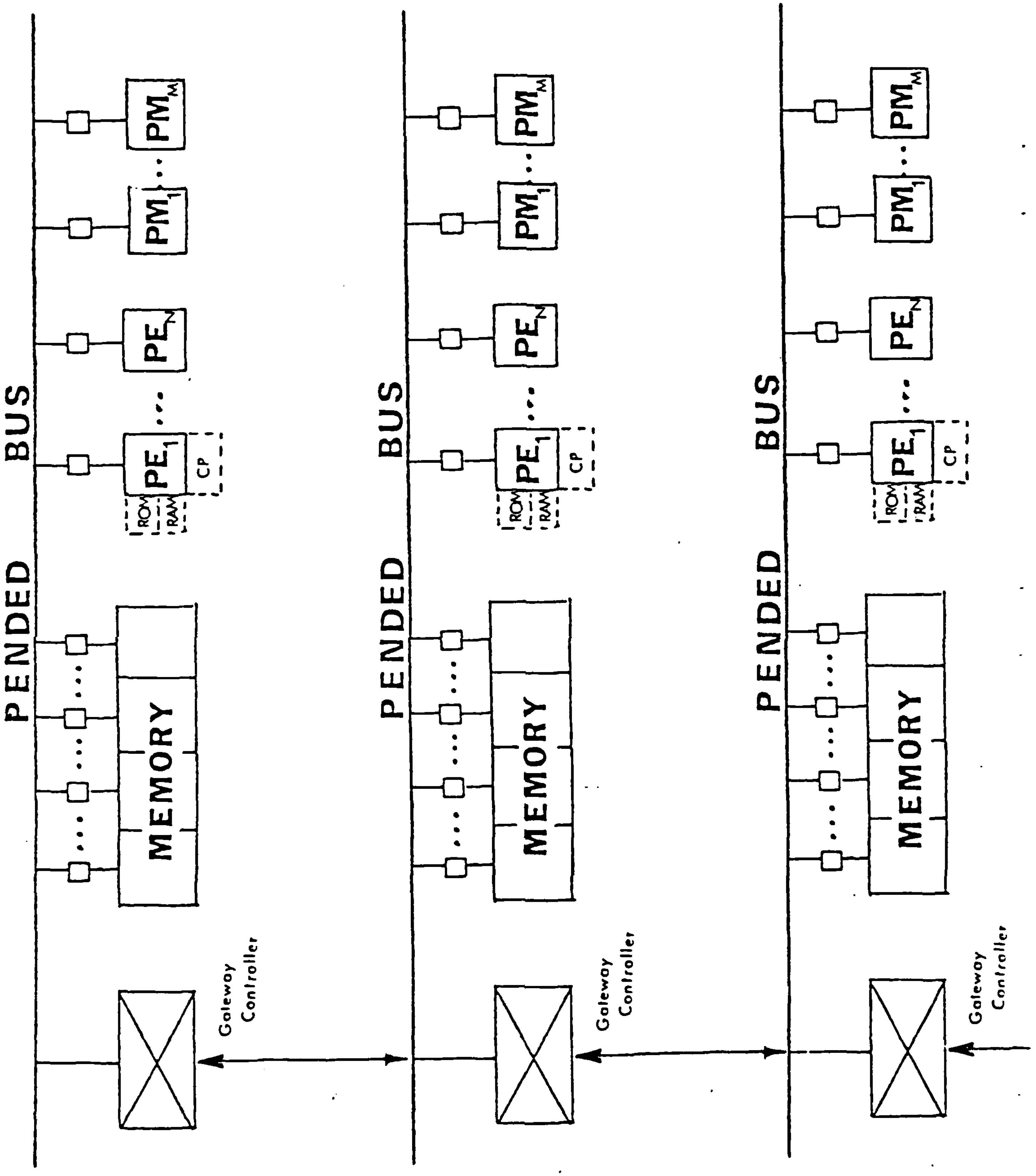
(e) Data manager aspect: It should support a dedicated data manager that will not be user programmable.

The overall effort is directed towards providing a high performance system using current generation, off-the-shelf components.

A block diagram of the proposed INFOPLEX hardware architecture is shown in Figure 4. The different sub-systems are described in the following paragraphs.

4.1 Multiprocessor Bus (MPB). The Multiprocessor Bus (MPB) is a high performance, high bandwidth, demand-multiplexed bus. Unlike conventional bus protocols where the bus is forced to remain idle, but unavailable for other work for large chunks of time, MPB facilitates full use of entire bandwidth for useful work. The MPB protocol splits the transaction into two parts, which use the bus in a nonconsecutive sequence; the bus is available for other work during th. intervening time duration.

The architecture of MPB introduces several innovations that further enhance its useful bandwidth. For example, in conventional structures, when two or more modules make simultaneous requests for bus usage, either one or no module is granted access to the bus; the others must make repeated requests. From modeling and simulation studies, it has been analytically demonstrated that an efficient arbitration mechanism, overlapped in time over bus usage and coupled with mechanisms for stacking of requests for bus usage, can reduce the percentage of refused requests to an insignificant percentage. Mathematically, if the queue capacity is 64 times the expected queue


PENDED BUS
MEMORY
ROM
RAM
PE_1
CP
PE_N
PM_1
PM_M
Gateway Controller
PENDED BUS
MEMORY
ROM
RAM
PE_1
CP
PE_N
PM_1
PM_M
Gateway Controller
PENDED BUS
MEMORY
ROM
RAM
PE_1
CP
PE_N
PM_1
PM_M
Gateway Controller

length, this probability is of the order of 10^{-24}. And for well designed systems, expected queue sizes are expected to be one or two messages long, and buffers of 64 bytes to 128 bytes can be easily implemented using contemporary off-the-shelf hardware.

In designing and implementing the MPB, the primary emphasis will be on speed. A higher speed implies that for a given transaction, the bus utilization reduces, and hence a higher level of multiprocessing becomes feasible. To this end, the system should be implemented using ECL 100K type integrated circuits; such type circuits have very high speed gates (under 1 ns. gate delays), and high bus driving capability for party line communications. Further, two MPB buses should be used (instead of a single bus) to enable better error detection and error correction capabilities, and possibly increased bandwidth, too.

4.2 BIU Modules. The basic bus interface unit (BIU) has several pieces, including: a transmitter; a receiver; address translation; bus arbitration; queue manager; maintenance, diagnostic and fault handler; module interface; and FSM control.

4.3 Processor Modules. Based on the reasons outlined in Section 2 and Section 3, it is proposed to use processors from the MC68000 family. The choice between the MC68000, the MC68010 and the MC68020 will be made in the future, depending on the availability of the processors.

4.4 Memory Modules. It is proposed to use high performance memory modules with single error correction and multiple error detection capabilities.

5. Multiprocessor Operating System (MOS)

In contrast to uniprocessor based operating systems, a tightly coupled multiprocessor will require a synthesis of both traditional as well as new control program concepts. Many of the design concepts in MOS can be considered akin to constructing a virtual machine system and simply providing more than one real physical processor to run jobs on. Virtual machine systems such as VM/370 have proven workable within constraints of considerable system overhead and very limited numbers of processors.

On the other hand, a primary design objective of MOS is that any real processor may, at any time, be executing any portion of its operating system code, independent of any other processor (exclusive of critical sections on locked data bases). Thus, there exists no master-slave relationship between any of the processors, except where defined by the user environment. It is felt that this will promote parallelism in independent task execution by reducing bottlenecks, and improve reliability by not requiring any singular or unique processors. Order is maintained via central system tables through which all system resources are allocated. This approach of using identical processors in the system is consistent with the INFOPLEX design objective of supporting maximum level of concurrent processing.

The primary objectives of MOS are: Simplicity, Reliability, and Modularity. The simpler the software architecture design and its implementation, the better chance that the system as a whole will perform optimally. However, adding reliability will often reduce the overall simplicity of the system and sometimes restrict its modularity. An appropriate balance among these three design objectives must be maintained.

Modularity is desired to keep different portions of the operating system as separate modules, with identifiable interfaces for simple documentation and modification, and safer overall operation.

5.1 MOS Overview. A major trend in computer systems technology is the decreasing cost of hardware compared to software. It has become increasingly popular to increase hardware resources, and to underutilize hardware, solely to simplify software systems.

The operating system will be completely distributed. There will be no central processor controlling the resources of the system. All microprocessors will be able to run MOS. The main consideration is to keep contention for MOS to a minimum, since it is a shared resource that could have a major performance impact on the system.

A major case in point is usage of a processing (CPU) resource. The CPU used to be the most scarce system resource, and extensive efforts were made to utilize it efficiently. Even in the C.mmp/HYDRA system, the processors were multiprogrammed to improve their usage. Major difficulties in multiprogramming multiprocessor systems are time for context switches and load leveling. It is now recognized that full CPU utilization is no longer a necessity, in the light of cheaper hardware.

To this end, it has been decided that there should be a strict avoidance of multiprogramming of different job streams on the processors in the DOS system. Microprocessor CPUs are considered cheap enough to be provided as needed without significant additional system cost. It is the task of the operating system to be flexible enough to take advantage of the CPU power available.

A major advantage of this decision is the ability to consider CPUs as devices, allocated and released to and from jobs much as one would use a magtape drive. At the very least, the system response time improves largely as a result of removing job queueing and state change delays in job executions. Thus, though there is no attempt made to salvage lost CPU power during I/O processing time, there is a corresponding decrease in the response time of a processor to an interrupt, and a decrease in the system overhead by the removal of complicated job schedulers and processor state changes.

Jobs in the system are organized into a parent/child tree, with the major system job at the root. As jobs enter the system, they form new branches, which may form sub-branches. The branches are pruned as jobs leave the system. This orderly job arrangement is a result of the fact that jobs must come from other jobs (such as a system job controlling a terminal), and not simply from nowhere. Parent and child processes can communicate, and parents are in control of their child processes. Input and output devices may be owned by jobs, and interrupts and I/O communicate directly to the associated job. Some interrupts may cause the spawning of a sub-job to handle it, such as an interrupt to the operating system to start a user job.

Devices, in general, are handled much the same as they are in any multiprogrammed uniprocessor system, since their use is much the same regardless of the processor or process using them. Considerations of allocation, deallocation, and deadlock prevention apply.

Memory management is perhaps the most difficult task in a tightly coupled multiprocessing system. The complete sharing of the memory resourcc in this tightly-coupled system allows complete flexibility in the memory usage of any process, as well as complete sharing of code and data blocks for memory efficiencies and communications, especially for operating system purposes. However, this usage must be allocated in an orderly manner, and proper protections must be available. Since there is no common control of execution of the processes in memory, no assumptions can be made about the usages of memory at any one time. The major impact of this fact is in the inability of any processor to move or modify any memory other than that which belongs to the process it is running, and then only if it is not shared information. The implications of this strategy will be discussed later.

A further difficulty encountered with most current LSI microprocessors, and even most minicomputers, is the inability to interrupt an instruction in progress. That is, the processors recognize an interrupt only after the end of the current instruction. The impact of this is that the processor cannot be stopped during a read or a write that cannot be completed (due to a non-resident memory segment) to be started up again later when a dummy operand is available. An alternative would be to supply a dummy operand to the current instruction, and to rerun the whole instruction later, but modern instruction sets prevent this. Register-modifying and recursive instructions are difficult or impossible to 'back-up' for proper execution later. As an example, consider an 'AND' to a register. If 0 were supplied as the dummy operand, the register contents would be lost. The same situation occurs with an all 1's operand to an 'OR' operation. Sophisticated operating system software and external hardware knowledge of the current processor state would be required for proper implementation.

Newer versions of microprocessors such as the Motorola 68020 have the power of current mainframe architectures for instruction interrupting, and are capable of supporting a complete virtual memory system. Use of such a microprocessor will simplify the hardware design of INFOPLEX, and also permit better performance.

Two major characteristics are required for multiprocessor software implementation. The first requirement is for an atomic Test-And-Set (TAS) operation executable by any processor to lock critical databases during modifications. Some processors have an implementation of a TAS primitive for use in master-slave type systems, but none of them are directly applicable to a split transaction single time-shared bus multiprocessor system. Therefore, the system hardware must provide a new mechanism.

The second major requirement is for an inter-processor communication system. Such a system is needed to carry operating system messages between processors for processor start/stop operations, system control, and memory management communications. The system would be partially software for message formatting and control, and partially hardware to create an interrupt at the destination processor to handle the message. Both the atomic TAS instructions and the inter-processor communication systems must be carefully implemented to allow high performance bandwidth without sacrificing the quality of the database.

6. Conclusion

In order to attain major improvements (in terms of storage capacity and speed of data retrievals) over conventional database systems, INFOPLEX strives to attain maximum concurrency at all dimensions. This high level of concurrency motivates the use of efficient split transaction protocols in order to support the inherent communication load involved. The need for maximum concurrency and the need to support split transaction protocols in an efficient manner are two factors that mandate a very careful evaluation and selection of microprocessor chips.

Several contemporary microprocessors offer excellent technical characteristics including high performance. Some of these chips can be used as efficient building blocks in a multimicroprocessor system configuration. Use of off-the-shelf hardware results in lower costs and minimal time delays as compared to the design and use of customized hardware. Among the various candidates that could be used in the INFOPLEX system, the Motorola 68000 appears to be currently best suited for this project because of the following reasons:

(a) Its 32-bit internal structure providing for higher throughput;

(b) Its bus arbitration modules (BAMs), which provide support in global bus multiprocessor design;

(c) The availability of TAS (test and set) instructions;

(d) The availability of signals needed for arbitration purposes;

(e) Its support of string manipulation instructions which have the potential of reducing turnaround times in processing database applications;

(f) The availability of auxiliary support chips;

(g) Its wide acceptance in the engineering and database community as the processor of choice for advanced architectures. Consequently, a growing body of applications and systems software is rapidly becoming available.

Overall, this report has documented the feasibility, from the hardware viewpoint, of the INFOPLEX design strategy.

References

1. R.B. Haagens, A Bus Structure for Multi-Microprocessing, S.M. Thesis, E.E. Dept., M.I.T., Cambridge, Mass., 1978.

2. J.A. Harris and D.R. Smith, "Hierarchical multiprocessor organizations," in Proc. 4th Symp. on Comput. Arch., March 1977, pp. 41-48.

3. E. Horowitz and A. Zorat, "The Binary Tree as an Interconnection Network: Applications to Multiprocessor Systems and VLSI," IEEE Trans. Comput., Vol. C-30, No. 4, April 1981, pp. 247-253.

4. H.D. Toong, "Micro-star - A microprocessor controlled distributed minicomputer network," IEEE COMPCON 77, 1977, pp. 320-324.

5. F. P. Preparata and J. Vuillemin, "The Cube Connected Cycles: A Versatile Network for Parallel Computation," Comm. ACM, Vol. 24, No. 5, May 1981, pp. 300-309.

6. L. D. Wittie, "Communication Structures for Large Networks of Microcomputers," IEEE Trans. Comput., Vol. C-30, No. 4, April 1981, pp. 264-273.

7. L.D. Wittie, "Efficient message routing in mega-microcomputer networks," in Proc. 3rd Symp. on Comput. Arch., Jan. 1976, pp. 136-140.

8. J.R. Goodman and C.H. Sequin, "Hypertree: A Multiprocessor Interconnection Topology," IEEE Trans. Comput., Vol. C-30, No. 12, December 1981, pp. 923-933.

9. Raphael A. Finkel and Marvin H. Solomon, "Processor Interconnection Strategies," IEEE Trans. Comput., Vol. C-29, No. 5, May 1980, pp. 360-371.

10. S.B. Wu and M.T. Liu, "A Cluster Structure as an Interconnection Network for Large Multimicrocomputer Systems," IEEE Trans. Comput., Vol. C-30, No. 4, April 1981, pp. 254-264.

11. K.J. Thurber and G.M. Masson, Distributed-Processor Communication Architecture, Lexington Books, Lexington, Mass., 1979.